# TITANIC DATA STUDIO

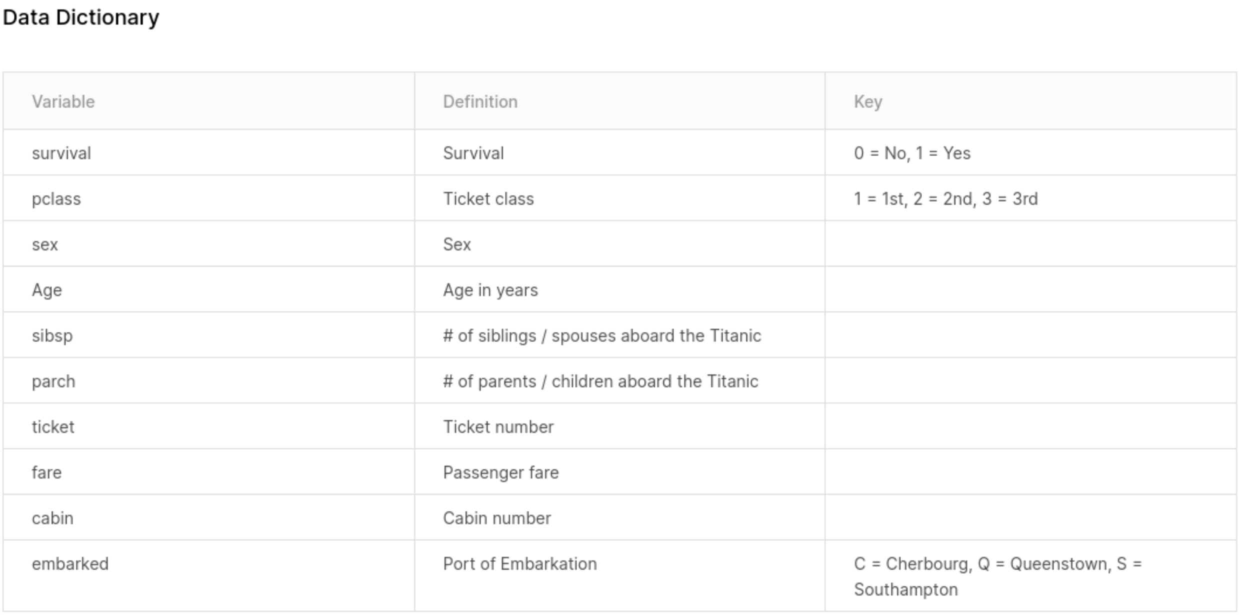
The objective of this study is discover and contribute whit new data to the tragedy of the titanic

the code will be commented and explained, feel free to manipulate and use

In [1]:

```
#import libraries and packages needed for the project

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn.model_selection as ms
from IPython.display import Image
from IPython.core.display import HTML
import re
import seaborn as sns
import plotly.express as px
from plotly.subplots import make_subplots

#image of data dictionary ( project description) , remember change the path to the image file

Image(url= "/home/dm/Desktop/titanic/g.png")
```

Out[1]:



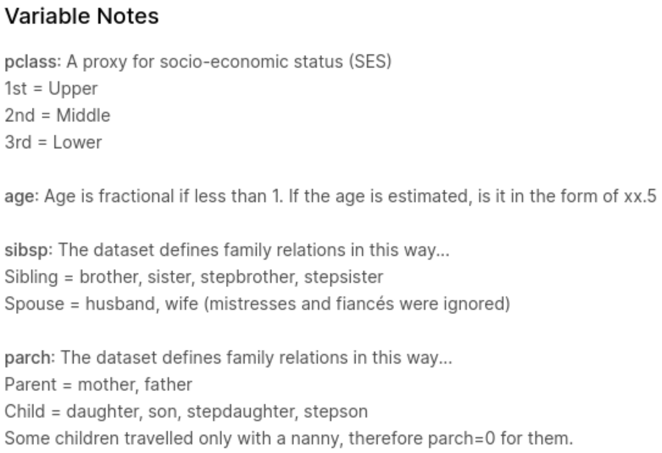
## Data Dictionary

| Variable | Definition | Key |
|---|---|---|
| survival | Survival | 0 = No, 1 = Yes |
| pclass | Ticket class | 1 = 1st, 2 = 2nd, 3 = 3rd |
| sex | Sex | |
| Age | Age in years | |
| sibsp | # of siblings / spouses aboard the Titanic | |
| parch | # of parents / children aboard the Titanic | |
| ticket | Ticket number | |
| fare | Passenger fare | |
| cabin | Cabin number | |
| embarked | Port of Embarkation | C = Cherbourg, Q = Queenstown, S = Southampton |

In [2]:

```
#second image of data guide, remember change the path to the image file

Image(url= "/home/dm/Desktop/titanic/v.png")
```

Out[2]:



## Variable Notes

**pclass**: A proxy for socio-economic status (SES)
1st = Upper
2nd = Middle
3rd = Lower

**age**: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

**sibsp**: The dataset defines family relations in this way...
Sibling = brother, sister, stepbrother, stepsister
Spouse = husband, wife (mistresses and fiancés were ignored)

**parch**: The dataset defines family relations in this way...
Parent = mother, father
Child = daughter, son, stepdaughter, stepson
Some children travelled only with a nanny, therefore parch=0 for them.

In [3]:

```
#first step is to load and check data

train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [4]:

```
#Check the train data

pd.DataFrame(train).head()
```

Out[4]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

In [5]:

```
#Check the test data

pd.DataFrame(test).head()
```

Out[5]:

| | PassengerId | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 892 | 3 | Kelly, Mr. James | male | 34.5 | 0 | 0 | 330911 | 7.8292 | NaN | Q |
| **1** | 893 | 3 | Wilkes, Mrs. James (Ellen Needs) | female | 47.0 | 1 | 0 | 363272 | 7.0000 | NaN | S |
| **2** | 894 | 2 | Myles, Mr. Thomas Francis | male | 62.0 | 0 | 0 | 240276 | 9.6875 | NaN | Q |
| **3** | 895 | 3 | Wirz, Mr. Albert | male | 27.0 | 0 | 0 | 315154 | 8.6625 | NaN | S |
| **4** | 896 | 3 | Hirvonen, Mrs. Alexander (Helga E Lindqvist) | female | 22.0 | 1 | 1 | 3101298 | 12.2875 | NaN | S |

In [6]:

```
# append the test data to the train data to get the complete data set

titanic_df = train.append(test, ignore_index=True)
```

```
<ipython-input-6-7b6b3afd4d7c>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
  titanic_df = train.append(test, ignore_index=True)
```

In [7]:

```
#check the type of the data to know and starting thinking ideas to apply the model

titanic_df.dtypes
```

Out[7]:

```
PassengerId      int64
Survived       float64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [8]:

```
#fist we need to normalize columns
#change the name of the columns to upper case and clean the data columns

titanic_df = titanic_df.rename(columns= lambda x: x.strip().replace(' ', '_').upper())
```

In [9]:

```
#check the data again to see the columns and in upper case and dont have any space or character

titanic_df.head()
```

Out[9]:

| | PASSENGERID | SURVIVED | PCLASS | NAME | SEX | AGE | SIBSP | PARCH | TICKET | FARE | CABIN | EMBARKED |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0.0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | S |
| **1** | 2 | 1.0 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| **2** | 3 | 1.0 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1.0 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0.0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

In [10]:

```
#this part is important make atention , we can see in the column "NAME" that there are titles in the name
#we need to clean the data and remove the titles in "NAME" column and group in anny better, lets go

"""
This function takes in a passenger's name and returns a string containing the title
if the Title is found, extract and return the title. If no title is found, return


"""

def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)
    if title_search:
        return title_search.group(1)
    return

#applying get_TITLE function to extract TITLE from name

titanic_df['TITLE'] = titanic_df['NAME'].apply(get_title)

 #printing unique values of TITLE column

titanic_df['TITLE'].unique()
```

Out[10]:

```
array(['Mr', 'Mrs', 'Miss', 'Master', 'Don', 'Rev', 'Dr', 'Mme', 'Ms',
       'Major', 'Lady', 'Sir', 'Mlle', 'Col', 'Capt', 'Countess',
       'Jonkheer', 'Dona'], dtype=object)
```

## TITLES

Young: boys and youngs under age

Don/Donna/Lady/Sir/Countess/Jonkheer: royal TITLEs

Rev: priest

Mme/Ms: single people

Major/Col/Capt: military

Mlle: married woman

In [11]:

```
#replacing master whit young
titanic_df['TITLE'] = titanic_df['TITLE'].replace(['Master'],'Young')

#replacing 'Ms' and 'Mlle' with 'Miss'
titanic_df['TITLE'] = titanic_df['TITLE'].replace(['Ms','Mlle'],'Miss')

#replacing 'Mme' with 'Mrs'
titanic_df['TITLE'] = titanic_df['TITLE'].replace('Mme','Mrs')

#replacing 'Don' with 'Royal'
titanic_df['TITLE'] = titanic_df['TITLE'].replace(['Don','Dona','Lady','Sir','Countess','Jonkheer'],'Royal')

 #replacing 'Major' and 'Col' with 'Military'
titanic_df['TITLE'] = titanic_df['TITLE'].replace(['Major','Col','Capt'],'Military')

#replacing 'Rev' with 'Priest'
titanic_df['TITLE'] = titanic_df['TITLE'].replace(['Rev'],'Priest')

#replacing 'Dr' with 'Medical'
titanic_df['TITLE'] = titanic_df['TITLE'].replace(['Dr'],'Medical')

#printing unique values of TITLE column
titanic_df['TITLE'].unique()
```

Out[11]:

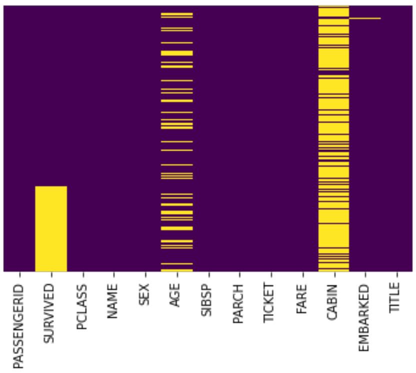
```
array(['Mr', 'Mrs', 'Miss', 'Young', 'Royal', 'Priest', 'Medical',
       'Military'], dtype=object)
```

In [12]:

```
#we can use to check empy or nulled data inside the columns the function heatmap
sns.heatmap(titanic_df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[12]:

```
<AxesSubplot:>
```



In [13]:

```
#first we want change nun to 0 in age variables of titanic_df
titanic_df["AGE"].fillna(0, inplace=True)

#replacing missing values of age with median age of each TITLE is better than common median model because it is more robust
def impute_age(row):

    # Features from row
    pclass = row['PCLASS']
    title = row['TITLE']
    age = row['AGE']

    if age == 0:
        return int(round(titanic_df.loc[(titanic_df['AGE']!=0)&
                                        (titanic_df['PCLASS']==pclass)&
                                        (titanic_df['TITLE']==title)]['AGE'].mean(),1))
    else:
        return age

titanic_df['AGE'] = titanic_df.apply(impute_age,axis=1)
```

In [14]:

```
#first we want change nun to 0 in age variables of titanic_df
titanic_df["AGE"].fillna(0, inplace=True)


#replacing missing values of age with median age of each TITLE is better than common median model because it is more robust
def impute_age(row):

    # Features from row
    pclass = row['PCLASS']
    title = row['TITLE']
    age = row['AGE']

    if age == 0:
        return int(round(titanic_df.loc[(titanic_df['AGE']!=0)&
                                      (titanic_df['PCLASS']==pclass)&
                                      (titanic_df['TITLE']==title)]['AGE'].mean(),1))
    else:
        return age

titanic_df['AGE'] = titanic_df.apply(impute_age,axis=1)
```

In [15]:

```
# Is posible to group the age data to use in future models

# number of age groups we want to split the data
splits = 8

# age range for each split
for i in range(splits):
    print(f'Group {i+1}:',pd.cut(titanic_df['AGE'].dropna(), splits).unique()[i])


# erase the age column and replace it with the age groups
titanic_df['AGE_GROUP'] = pd.cut(titanic_df['AGE'].dropna(), splits)

#check our age groups
group_age = titanic_df.groupby('AGE_GROUP')['AGE']
```

```
Group 1: (20.128, 30.106]
Group 2: (30.106, 40.085]
Group 3: (50.064, 60.043]
Group 4: (0.0902, 10.149]
Group 5: (10.149, 20.128]
Group 6: (40.085, 50.064]
Group 7: (60.043, 70.021]
Group 8: (70.021, 80.0]
```

In [16]:

```
 #are im sure there are no duplicates in titanic_df?

titanic_df_duplicates = titanic_df.duplicated()
print('Number of duplicate entries is/are {}'.format(titanic_df_duplicates.sum()))
```

```
Number of duplicate entries is/are 0
```

In [17]:

```
#survived column is missing, we need to fill it using median

titanic_df['SURVIVED'].fillna(titanic_df['SURVIVED'].median(), inplace=True)
```

In [18]:

```
#droping the columns that we don't need anymore

titanic_df = titanic_df.drop(["CABIN"] , axis=1)
titanic_df = titanic_df.drop(["PASSENGERID"] , axis=1)
```

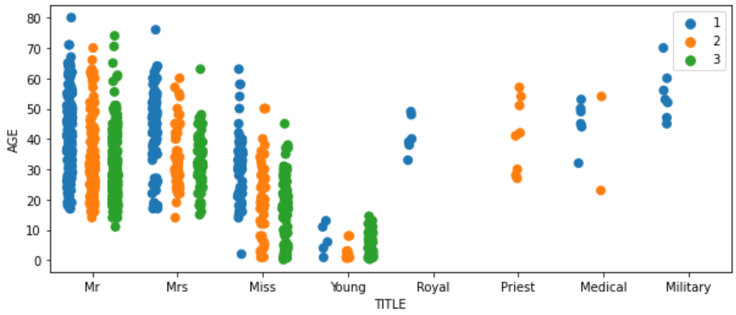
In [19]:

```
titanic_df["EMBARKED"] = titanic_df["EMBARKED"].fillna("S") #filling the missing values with S
```

In [20]:

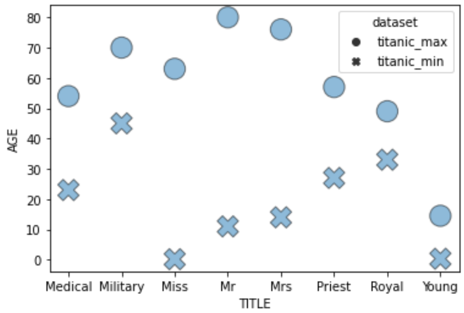
```
#visualizing the data comparing the AGE , TITLE and PCLASS columns

plt.figure(figsize=(10,4))
sns.stripplot(x='TITLE',y='AGE',data=titanic_df[titanic_df['AGE']!=0], hue='PCLASS',dodge=True, size=8)

plt.legend(loc=1)
print("We can see how the pclass and title affect the age of the passengers")
```

```
We can see how the pclass and title affect the age of the passengers
```



In [21]:

```
#creating a dataframe of max age of each TITLE
titanic_max = pd.DataFrame(titanic_df.groupby('TITLE')['AGE'].max())

#creating a dataframe of min age of each TITLE
titanic_min = pd.DataFrame(titanic_df.groupby('TITLE')['AGE'].min())

#concatenating titanic_max and titanic_min dataframes
titanic_max_min = pd.concat([titanic_max.assign(dataset='titanic_max'), titanic_min.assign(dataset='titanic_min')])

#plotting scatterplot of max and min age of each TITLE
sns.scatterplot(x='TITLE', y='AGE', data=titanic_max_min, style='dataset', palette='Set1', s=300, alpha=0.5, linewidth=1, edgecolor='blac
k', )

print(plt.show(), titanic_max_min)
print("See how maximum and minimum age of each title affect the age of the passengers")
```



```
None             AGE      dataset
TITLE
Medical   54.00  titanic_max
Military  70.00  titanic_max
Miss      63.00  titanic_max
Mr        80.00  titanic_max
Mrs       76.00  titanic_max
Priest    57.00  titanic_max
Royal     49.00  titanic_max
Young     14.50  titanic_max
Medical   23.00  titanic_min
Military  45.00  titanic_min
Miss       0.17  titanic_min
Mr        11.00  titanic_min
Mrs       14.00  titanic_min
Priest    27.00  titanic_min
Royal     33.00  titanic_min
Young      0.33  titanic_min
See how maximum and minimum age of each title affect the age of the passengers
```

In [22]:

```
#creating a dataframe of mean age of each TITLE
mean_age_title = titanic_df.groupby('TITLE')['AGE'].mean()
mean_age_title = pd.DataFrame(mean_age_title, columns=['AGE'])
mean_age_title
```

Out[22]:

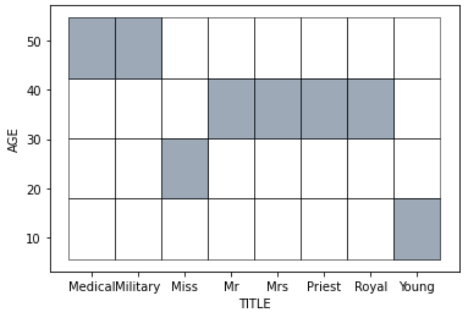
| | AGE |
|---|---|
| **TITLE** | |
| **Medical** | 43.750000 |
| **Military** | 54.714286 |
| **Miss** | 20.964356 |
| **Mr** | 31.795905 |
| **Mrs** | 36.757576 |
| **Priest** | 41.250000 |
| **Royal** | 41.166667 |
| **Young** | 5.550492 |

In [23]:

```
#plot of mean age of each TITLE encapuslated in a age square whit ten years size
sns.histplot(x='TITLE', y='AGE', data=mean_age_title,  palette='Set1', alpha=0.5, linewidth=1, edgecolor='black', ) #plotting histogram o
f mean age of each TITLE
```

Out[23]:

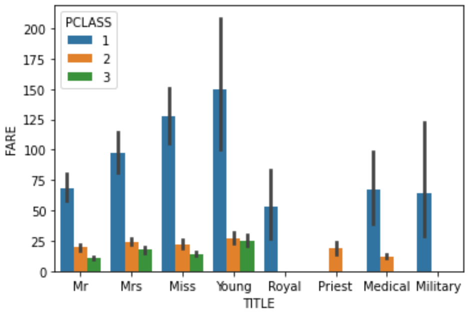
```
<AxesSubplot:xlabel='TITLE', ylabel='AGE'>
```



In [24]:

```
 #plotting scatterplot of fare vs title, remember "FARE" is equal to Ticket Price in Dollars

sns.barplot(x='TITLE', y='FARE', data=titanic_df, hue='PCLASS' )
```

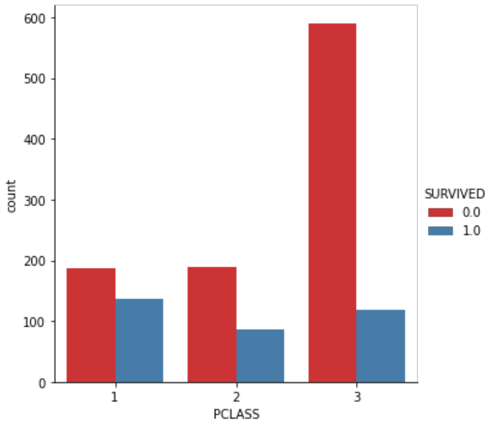
Out[24]: `<AxesSubplot:xlabel='TITLE', ylabel='FARE'>`



In [25]:

```
#check the survival of each title

sns.catplot(x='PCLASS',  data=titanic_df, hue='SURVIVED', kind='count', palette='Set1')
```

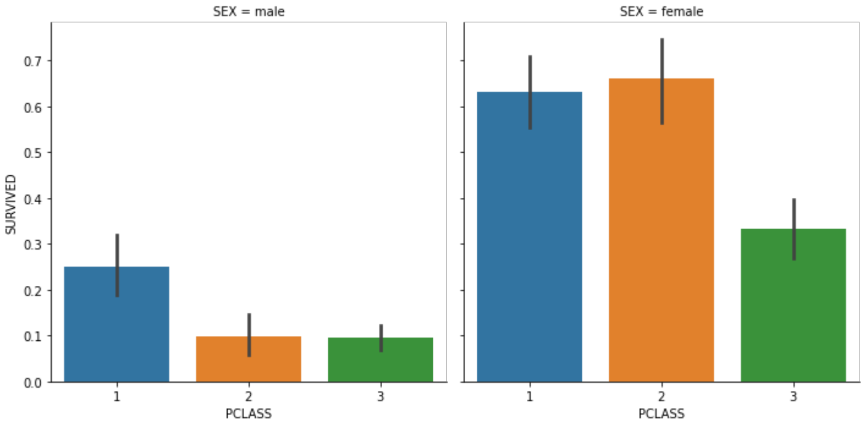
Out[25]: `<seaborn.axisgrid.FacetGrid at 0x7f2835d1b6d0>`



In [26]:

```
#passanger class survival distribution by sex

sns.catplot(x='PCLASS', y='SURVIVED', kind='bar', data=titanic_df, col='SEX')
```

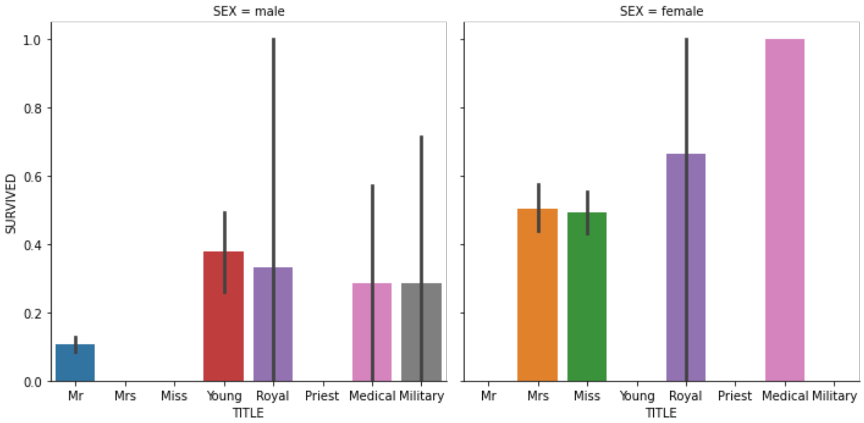
Out[26]: `<seaborn.axisgrid.FacetGrid at 0x7f2835d11ac0>`



In [27]:

```
# dead sex distribution titles based

sns.catplot(x='TITLE', y='SURVIVED', kind='bar', data=titanic_df, col='SEX')
```

Out[27]: `<seaborn.axisgrid.FacetGrid at 0x7f2835cef130>`



In [28]:

```
#passenger survival rating

fig = px.pie(titanic_df, names='SURVIVED', title='Passenger Survival', hole=0.3)
fig.show()
```

In [29]:

```
#Class vs age data

fig = px.box(titanic_df, x='PCLASS', y="AGE", color='SURVIVED',)
fig.show()
```

In [30]:

```
#title vs age data

fig = px.box(titanic_df, x='TITLE', y="AGE", color='SURVIVED')
fig.show()
```

In [31]:

```
#Whats is the worst age to survive? (Yellow)

fig = px.density_heatmap(titanic_df, x="SURVIVED", y="AGE")
fig.show()
```

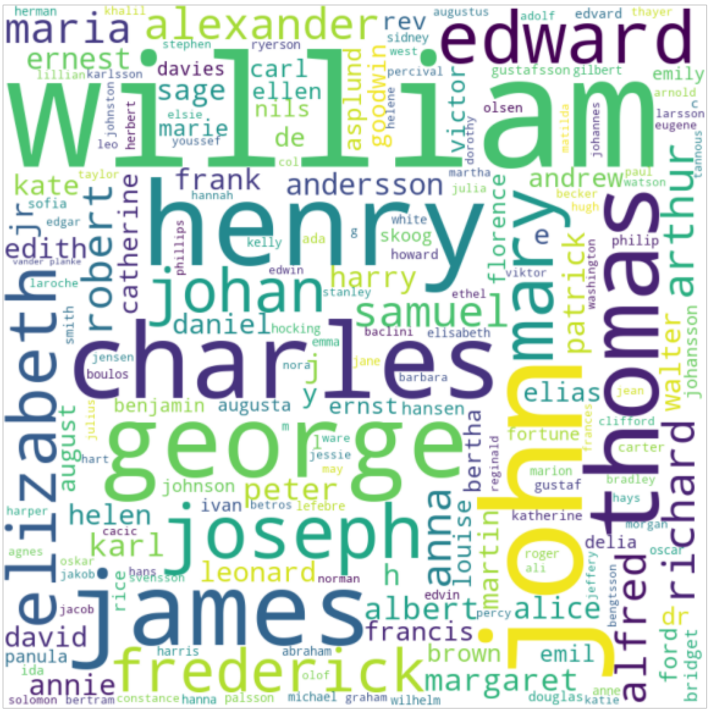
In [32]:

```
#What is the most commons names in the titanic dataset?

#creating a dataframe of most common names in the titanic dataset

titanic_words = titanic_df['NAME'].str.split(' ').tolist() #splitting name column into words
titanic_words = [item for sublist in titanic_words for item in sublist] #flattening the list
titanic_words = pd.DataFrame(titanic_words) #creating a dataframe of words
titanic_words.columns = ['NAME'] #renaming column
titanic_words['NAME'] = titanic_words['NAME'].str.replace('\.', '')  #removing dots from the words
titanic_words['NAME'] = titanic_words['NAME'].str.replace('"', ' ')  #removing quotes from the words
titanic_words['NAME'] = titanic_words['NAME'].str.replace(',', ' ')  #removing commas from the words
titanic_words = titanic_words[titanic_words["NAME"].str.contains("Mr|Mrs|Miss|Master|Royal|Priest|Medical|Military")==False] #removing Mr, Mrs, Miss, Master, Royal, Priest, Medical and Military from the words
```

```
<ipython-input-32-96dbb3c45195>:9: FutureWarning:

The default value of regex will change from True to False in a future version.
```

In [33]:

```
#we can response now this doing a cloud of the most common names in the titanic dataset

from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt


palabras = ''
stopwords = set(STOPWORDS)

for val in titanic_words.NAME:

    val = str(val)
    tokens = val.split()
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    palabras += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(palabras)

plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



In [34]:

```python
# starting charging the lifeboats of the titanic dataset

lifeboats = pd.read_csv('Lifeboats.csv')
lifeboats = lifeboats.drop(['Unnamed: 0'], axis=1) #removing the unamed column
lifeboats['launch_time'] = lifeboats['launch'].apply(lambda x: x.split(' ')[1])   #splitting the launch time into hours and minutes
lifeboats.drop(['launch'], axis=1, inplace=True) #removing the launch column

lifeboats
```

Out[34]:

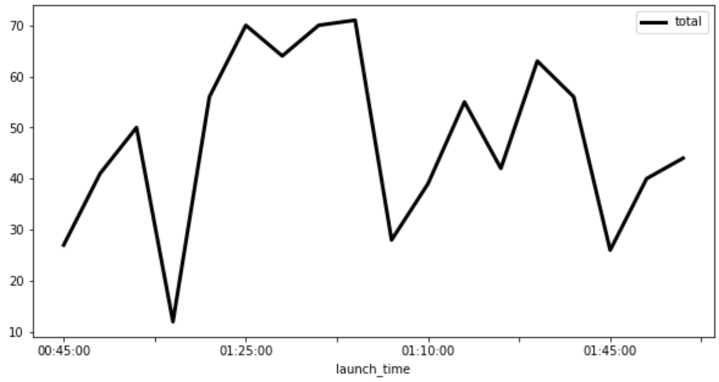
| | side | boat | crew | men | women | total | cap | launch_time |
|---|---|---|---|---|---|---|---|---|
| **0** | Port | 7 | 3 | 4 | 20 | 27 | 65 | 00:45:00 |
| **1** | Port | 5 | 5 | 6 | 30 | 41 | 65 | 00:55:00 |
| **2** | Port | 3 | 15 | 10 | 25 | 50 | 65 | 01:00:00 |
| **3** | Port | 1 | 7 | 3 | 2 | 12 | 40 | 01:10:00 |
| **4** | Port | 9 | 8 | 6 | 42 | 56 | 65 | 01:20:00 |
| **5** | Port | 11 | 9 | 1 | 60 | 70 | 65 | 01:25:00 |
| **6** | Port | 13 | 5 | 0 | 59 | 64 | 65 | 01:35:00 |
| **7** | Port | 15 | 13 | 4 | 53 | 70 | 65 | 01:35:00 |
| **8** | Port | C | 5 | 2 | 64 | 71 | 47 | 01:40:00 |
| **9** | Starboard | 6 | 2 | 2 | 24 | 28 | 65 | 00:55:00 |
| **10** | Starboard | 8 | 4 | 0 | 35 | 39 | 65 | 01:10:00 |
| **11** | Starboard | 10 | 5 | 0 | 50 | 55 | 65 | 01:20:00 |
| **12** | Starboard | 12 | 2 | 0 | 40 | 42 | 65 | 01:25:00 |
| **13** | Starboard | 14 | 8 | 2 | 53 | 63 | 65 | 01:30:00 |
| **14** | Starboard | 16 | 6 | 0 | 50 | 56 | 65 | 01:35:00 |
| **15** | Starboard | 2 | 4 | 1 | 21 | 26 | 40 | 01:45:00 |
| **16** | Starboard | 4 | 4 | 0 | 36 | 40 | 65 | 01:55:00 |
| **17** | Starboard | D | 2 | 2 | 40 | 44 | 47 | 02:05:00 |

In [35]:

```python
#how was the lifeboats launched?

lifeboats.plot(x='launch_time', y='total', kind='line', figsize=(10,5), color='#000000', linewidth=3)
```

Out[35]:

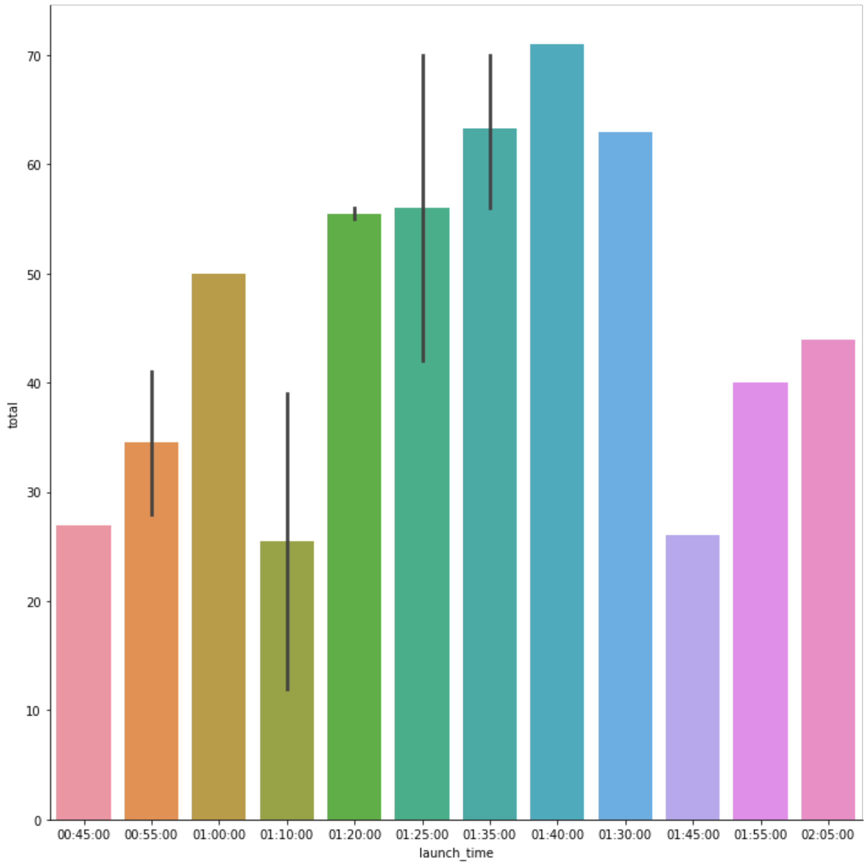
```
<AxesSubplot:xlabel='launch_time'>
```



```
In [36]: #how was the size of the lifeboats launched?

         sns.catplot(x='launch_time', y='total', kind='bar', data=lifeboats, size=10)
```

```
/home/dm/.local/lib/python3.8/site-packages/seaborn/categorical.py:3750: UserWarning:

The `size` parameter has been renamed to `height`; please update your code.
```

```
Out[36]: <seaborn.axisgrid.FacetGrid at 0x7f28348cfa30>
```



```
In [37]: lifeboats.value_counts()
```

```
Out[37]: side       boat  crew  men  women  total  cap  launch_time
         Port       1     7     3    2      12     40   01:10:00       1
                    11    9     1    60     70     65   01:25:00       1
         Starboard  8     4     0    35     39     65   01:10:00       1
                    6     2     2    24     28     65   00:55:00       1
                    4     4     0    36     40     65   01:55:00       1
                    2     4     1    21     26     40   01:45:00       1
                    16    6     0    50     56     65   01:35:00       1
                    14    8     2    53     63     65   01:30:00       1
                    12    2     0    40     42     65   01:25:00       1
                    10    5     0    50     55     65   01:20:00       1
         Port       C     5     2    64     71     47   01:40:00       1
                    9     8     6    42     56     65   01:20:00       1
                    7     3     4    20     27     65   00:45:00       1
                    5     5     6    30     41     65   00:55:00       1
                    3     15    10   25     50     65   01:00:00       1
                    15    13    4    53     70     65   01:35:00       1
                    13    5     0    59     64     65   01:35:00       1
         Starboard  D     2     2    40     44     47   02:05:00       1
         dtype: int64
```

```
In [38]: #3d plot of the lifeboats launched testing
         import plotly.express as px

         fig = px.scatter_3d(lifeboats, x='launch_time', y='total', z='cap', color='cap', size='cap', title='Lifeboats')
         fig.show()
```

```
In [39]: #how was evacuated?

         lifeboats.groupby('launch_time')['total'].sum()
```

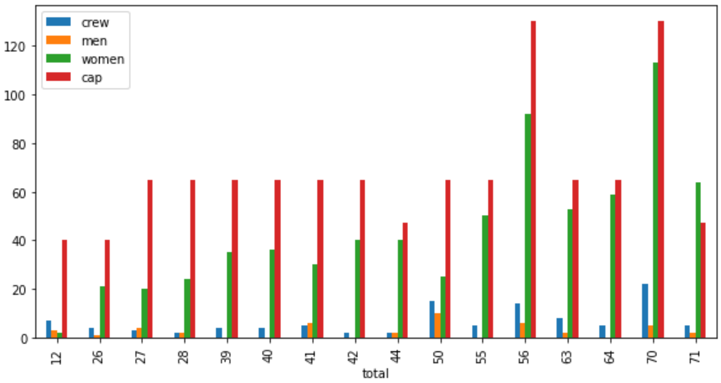
```
Out[39]: launch_time
         00:45:00     27
         00:55:00     69
         01:00:00     50
         01:10:00     51
         01:20:00    111
         01:25:00    112
         01:30:00     63
         01:35:00    190
         01:40:00     71
         01:45:00     26
         01:55:00     40
         02:05:00     44
         Name: total, dtype: int64
```

In [40]:

```
#how much people survived in boats?

lifeboats.groupby('total').sum().plot(kind='bar', figsize=(10,5),  linewidth=1 )
```

Out[40]:

```
<AxesSubplot:xlabel='total'>
```



In [41]:

```
#how much people will be evacuated if the boats are full?

lifeboats["cap"].sum()
```

Out[41]:

```
1084
```

In [42]:

```
#how much people was evacuated?

lifeboats.groupby('launch_time')['total'].sum()
```

Out[42]:

```
launch_time
00:45:00     27
00:55:00     69
01:00:00     50
01:10:00     51
01:20:00    111
01:25:00    112
01:30:00     63
01:35:00    190
01:40:00     71
01:45:00     26
01:55:00     40
02:05:00     44
Name: total, dtype: int64
```

hows much boats was necessary to evacuate all the people?

In [43]:

```
#fist we need to know the mean of the capacity of the boats
lifeboats_cap = lifeboats['cap'].mean()
lifeboats_cap
```

Out[43]:

```
60.22222222222222
```

In [44]:

```
#know first the total of people saved by the boats
titanic_df["SURVIVED"].sum()
```

Out[44]:

```
342.0
```

In [45]:

```
#know the total of people dead in the titanic
count = (titanic_df['SURVIVED'] == 0).sum()
count
```

Out[45]:

```
967
```

In [46]:

```
print("Titanic lifeboats necessary to save all people :", count / lifeboats_cap )
```

```
Titanic lifeboats necessary to save all people : 16.05719557195572
```

In [47]:

```
#here same solution but using a function, better solution :)
def lifeboats_necessary(titanic_df, lifeboats):
    lifeboats_cap = lifeboats['cap'].mean()
    titanic_df["SURVIVED"].sum()
    count = (titanic_df['SURVIVED'] == 0).sum()
    return count / lifeboats_cap

lifeboats_necessary(titanic_df, lifeboats)
```

Out[47]:

```
16.05719557195572
```

In [48]:

```
#The best side to take a boat
lifeboats_side = lifeboats.groupby('side')['total', "cap"].sum()
lifeboats_side
```

```
<ipython-input-48-a938a8115936>:2: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

Out[48]:

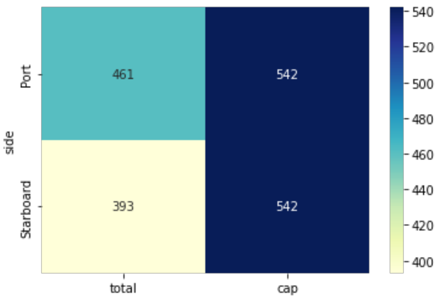
| | total | cap |
|---|---|---|
| **side** | | |
| **Port** | 461 | 542 |
| **Starboard** | 393 | 542 |

In [49]:

```python
sns.heatmap(lifeboats_side, annot=True, fmt='.0f', cmap='YlGnBu')
```

Out[49]:

```
<AxesSubplot:ylabel='side'>
```



In [50]:

```python
lifeboats
```

Out[50]:

| | side | boat | crew | men | women | total | cap | launch_time |
|---|---|---|---|---|---|---|---|---|
| **0** | Port | 7 | 3 | 4 | 20 | 27 | 65 | 00:45:00 |
| **1** | Port | 5 | 5 | 6 | 30 | 41 | 65 | 00:55:00 |
| **2** | Port | 3 | 15 | 10 | 25 | 50 | 65 | 01:00:00 |
| **3** | Port | 1 | 7 | 3 | 2 | 12 | 40 | 01:10:00 |
| **4** | Port | 9 | 8 | 6 | 42 | 56 | 65 | 01:20:00 |
| **5** | Port | 11 | 9 | 1 | 60 | 70 | 65 | 01:25:00 |
| **6** | Port | 13 | 5 | 0 | 59 | 64 | 65 | 01:35:00 |
| **7** | Port | 15 | 13 | 4 | 53 | 70 | 65 | 01:35:00 |
| **8** | Port | C | 5 | 2 | 64 | 71 | 47 | 01:40:00 |
| **9** | Starboard | 6 | 2 | 2 | 24 | 28 | 65 | 00:55:00 |
| **10** | Starboard | 8 | 4 | 0 | 35 | 39 | 65 | 01:10:00 |
| **11** | Starboard | 10 | 5 | 0 | 50 | 55 | 65 | 01:20:00 |
| **12** | Starboard | 12 | 2 | 0 | 40 | 42 | 65 | 01:25:00 |
| **13** | Starboard | 14 | 8 | 2 | 53 | 63 | 65 | 01:30:00 |
| **14** | Starboard | 16 | 6 | 0 | 50 | 56 | 65 | 01:35:00 |
| **15** | Starboard | 2 | 4 | 1 | 21 | 26 | 40 | 01:45:00 |
| **16** | Starboard | 4 | 4 | 0 | 36 | 40 | 65 | 01:55:00 |
| **17** | Starboard | D | 2 | 2 | 40 | 44 | 47 | 02:05:00 |

In [51]:

```python
# can know the evacuation hours of the boats?
px.bar(lifeboats, x='cap', y='launch_time', color='side', title='Lifeboats by side')
```

In [52]:

```python
import os
os.system('jupyter nbconvert --execute --to html titanic.ipynb')
```

Out[52]:

```
256
```